

---

# MONGODB - NoSQL CRUD

---

— Phó Nghĩa Văn —



**CYBERSOFT**  
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH



**MYC**  **DER**

**timviec**  **it.com**

# MỤC TIÊU BÀI HỌC

- Nắm được khái niệm hệ quản trị cơ sở dữ liệu MongoDB
- Hiểu sự khác nhau cơ bản giữa NoSQL và SQL
- Cài đặt được mongoDB
- Kết nối được với mongoDB, tạo schema/model, tạo instance
- Import data.json vào mongoDB
- Thực hiện các lệnh truy vấn
- CRUD với mongoDB (create, read, update, delete/destroy)

# GIỚI THIỆU VỀ MONGODB

**MongoDB** là một hệ quản trị cơ sở dữ liệu mã nguồn mở được thiết kế theo kiểu hướng đối tượng trong đó các bảng được cấu trúc một cách linh hoạt cho phép các dữ liệu lưu trên bảng không cần phải tuân theo một dạng cấu trúc nhất định nào. Chính do cấu trúc linh hoạt này nên MongoDB có thể được dùng để lưu trữ các dữ liệu có cấu trúc phức tạp và đa dạng và không cố định (hay còn gọi là Big Data).



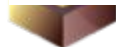
# ƯU ĐIỂM MONGODB

- Ít schema hơn
- Cấu trúc một đối tượng rõ ràng
- Không có các join phức tạp
- Khả năng truy vấn sâu, đa dạng hơn
- Dễ dàng mở rộng
- Kho lưu định hướng Document: Dữ liệu được lưu trong các tài liệu kiểu JSON
- Cập nhật nhanh hơn

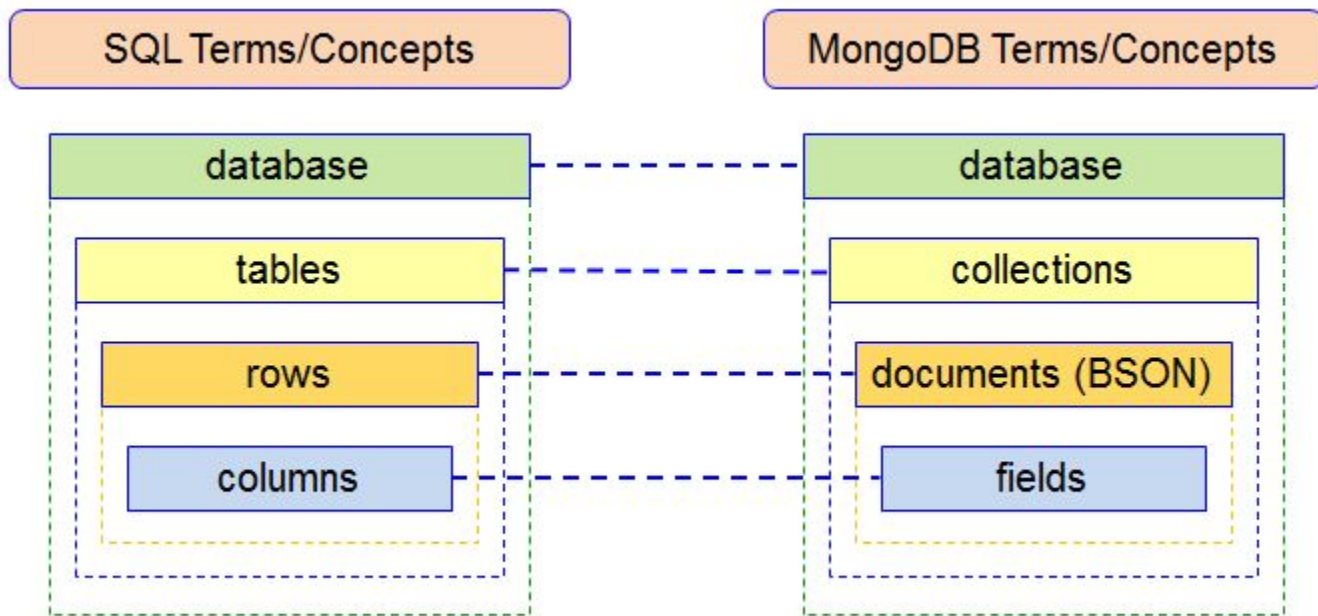


# SO SÁNH SQL VÀ NoSQL

	SQL	NoSQL
Type	Relational	Non-Relational
Data	Structured Data stored in Tables	Un-structured stored in JSON files but the graph database does supports relationship
Schema	Static	Dynamic
Scalability	Vertical	Horizontal
Language	Structured Query Language	Un-structured Query Language
Joins	Helpful to design complex queries	No joins, Don't have the powerful interface to prepare complex query
OLTP	Recommended and best suited for OLTP systems	Less likely to be considered for OLTP system
Support	Great support	community depedent, they are expanding the support model
Integrated Caching	Supports In-line memory(SQL2014 and SQL 2016)	Supports integrated caching
flexible	rigid schema bound to relationship	Non-rigid schema and flexible
Transaction	ACID	CAP theorem
Auto elasticity	Requires downtime in most cases	Automatic, No outage required



# SO SÁNH SQL VÀ NoSQL



# SO SÁNH SQL VÀ NoSQL

## RELATIONAL

Posts (id, Title)

1	Title
---	-------

Comments

01	1	Comment 1
02	1	Comment 2

## NON-RELATIONAL

Posts (id, Title, Comments / Image)

1	Title	Comment 1
		Comment 2
		Comment 3
<hr/>		
2	Title 2	Image



# CÔNG CỤ

- MongoDB: <https://www.mongodb.com/>
- MongoDB local:
  - Robomongo: <https://robomongo.org/>
  - Compass: <https://www.mongodb.com/products/compass>
- MongoDB cloud:
  - mlab: <https://mlab.com/>
  - atlas: <https://cloud.mongodb.com>

Trong bài này, chúng ta sẽ dùng **Robomongo**



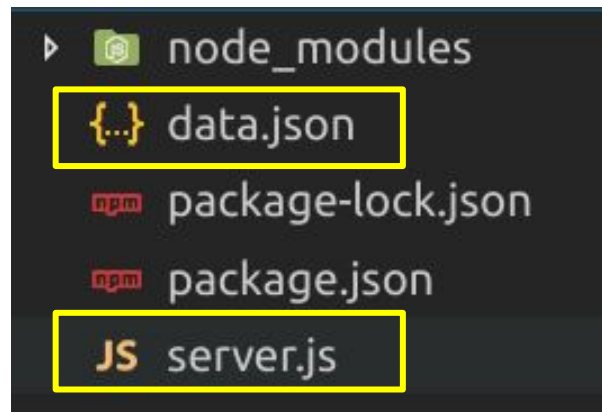


**HAPPY CODING :)**



# CONNECT MONGODB

- `npm init -y`
- `npm install mongoose nodemon`
- Tạo file **server.js**
- File **data.json** được cung cấp từ LMS

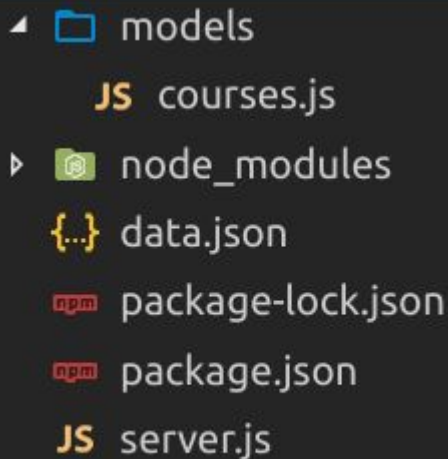


```
const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/courses')
  .then(() => console.log("Connected to MongoDB"))
  .catch(err => console.log("Error", err))
```

# SCHEMA - MODEL

Tạo Schema  
và model  
như hình bên



File explorer showing project structure:

- models
  - courses.js
- node\_modules
- data.json
- package-lock.json
- package.json
- server.js

```
const mongoose = require('mongoose');

const CourseSchema = new mongoose.Schema({
  name: String,
  author: String,
  tags: [String],
  date: {type: Date, default: Date.now},
  isPublished: Boolean,
  price: Number
})

const Course = mongoose.model('courses', CourseSchema);
module.exports = Course;
```

# CREATE AND SAVE DOCUMENT

```
const mongoose = require('mongoose');  
const Course = require('./models/courses');
```

import Course model

```
mongoose.connect('mongodb://localhost:27017/courses')  
  .then(() => console.log("Connected to MongoDB"))  
  .catch(err => console.log("Error", err))
```

```
const newCourse = new Course({  
  name: "FullstackJS",  
  author: "Hackagon",  
  tags: ["ReactJS", "MongoDB", "NodeJS"],  
  isPublished: true,  
  price: 300  
})
```

Tạo mới một  
document/instance

Save document

```
newCourse.save()  
  .then(course => console.log(JSON.stringify(course, undefined, 2)))
```



# KẾT QUẢ

- Chạy lệnh **node server.js**
- Kiểm tra kết quả tại terminal và robomongo

The screenshot shows the RoboMongo interface with a new connection to localhost:27017. The 'courses' collection is selected, and a query is executed. The result is a single document with the following fields:

Key	Value
(1) ObjectId("5bf52a79e9c4dd1ff6d9eec3")	{ 8 fields }
_id	ObjectId("5bf52a79e9c4dd1ff6d9eec3")
tags	[ 3 elements ]
name	FullstackJS
author	Hackagon
isPublished	true
price	300
date	2018-11-21 09:50:49.327Z
__v	0

```
{
  "tags": [
    "ReactJS",
    "MongoDB",
    "NodeJS"
  ],
  "_id": "5bf52ccb64fb5b227853bc09",
  "name": "FullstackJS",
  "author": "Hackagon",
  "isPublished": true,
  "price": 300,
  "date": "2018-11-21T10:00:43.977Z",
  "__v": 0
}
```

# ObjectId

- ObjectId là key mặc định cho một document trong MongoDB và có thể được tìm thông qua **\_id**
- ObjectId is a 12 byte binary BSON type:
- Sử dụng kèm với các phương thức: **findById()**, **findByIdAndUpdate()**, **findByIdAndRemove()**

Size	Description
4 bytes	a 4-byte value representing the seconds since the Unix epoch
3 bytes	a 3-byte machine identifier
2 bytes	2-byte process id
3 bytes	3-byte counter, starting with a random value



# IMPORT DATA

- Mở mongoDB service
- Gõ lệnh **mongoimport --db courses --collection courses --file data.json --jsonArray** trong terminal

```
2018-11-21T23:05:39.401+0700    connected to: localhost
2018-11-21T23:05:39.421+0700    imported 7 documents
```

```
db.getCollection('courses').find({})
```

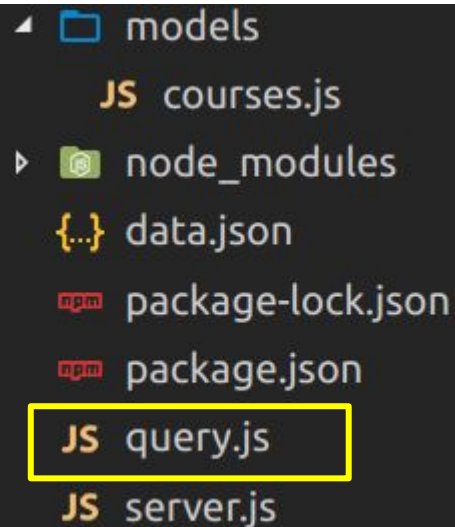
courses 0.001 sec.

Key	Value
▶ (1) 5a68fdd7bee8ea64649c2777	{ 8 fields }
▶ (2) 5a68fde3f09ad7646ddec17e	{ 8 fields }
▶ (3) 5a68fe2142ae6a6482c4c9cb	{ 8 fields }
▶ (4) 5a68fdf95db93f6477053ddd	{ 7 fields }
▶ (5) 5a68ff090c553064a218a547	{ 8 fields }
▶ (6) 5a6900fff467be65019a9001	{ 8 fields }
▶ (7) 5a68fdc3615eda645bc6bdec	{ 8 fields }



# QUERY DOCUMENTS

- Tạo file **query.js**
- Thực hiện connect đến MongoDB trong file này (tương tự như server.js)



```
const mongoose = require('mongoose');
const Course = require('./models/courses');

mongoose.connect('mongodb://localhost:27017/courses', { useNewUrlParser: true })
  .then(() => console.log("Connected to MongoDB"))
  .catch(err => console.log("Error", err))
```



# QUERY DOCUMENTS

- **find()**: hiển thị tất cả các document (không truyền vào bất kỳ tham số nào)

```
Course.find()  
  .then(courses => console.log(courses))
```

- Chạy lệnh **nodemon server.js** --> Hiển thị ra tất cả các documents



```
[ { tags: [ 'node', 'backend' ],  
  _id: 5a68fdd7bee8ea64649c2777,  
  date: 2018-01-24T21:42:47.912Z,  
  name: 'NodeJS',  
  author: 'Mosh',  
  isPublished: true,  
  price: 20,  
  __v: 0 },  
  { tags: [ 'aspnet', 'backend' ],  
    _id: 5a68fde3f09ad7646ddec17e,  
    date: 2018-01-24T21:42:59.605Z,  
    name: 'ASP.NET MVC',  
    author: 'Mosh',  
    isPublished: true,  
    price: 15,  
    __v: 0 },  
  { tags: [ 'node', 'backend' ],  
    _id: 5a68fe2142ae6a6482c4c9cb,  
    date: 2018-01-24T21:44:01.075Z,  
    name: 'NodeJS by Jack',  
    author: 'Jack',  
    isPublished: true,  
    price: 12,  
    __v: 0 },  
  { tags: [ 'react', 'frontend' ],  
    _id: 5a68fdf95db93f6477053ddd,  
    date: 2018-01-24T21:43:21.589Z,  
    name: 'React',  
    author: 'Mosh',  
    isPublished: false,  
    __v: 0 },
```

# QUERY DOCUMENTS

- **find()**: với tham số là object
- **limit()**: giới hạn số documents
- **select()**: chọn field hiển thị

```
[ { tags: [ 'aspnet', 'backend' ],  
  _id: 5a68fde3f09ad7646ddec17e,  
  name: 'ASP.NET MVC' },  
  { tags: [ 'angular', 'frontend' ],  
    _id: 5a6900fff467be65019a9001,  
    name: 'Angular' },  
  { tags: [ 'express', 'backend' ],  
    _id: 5a68fdc3615eda645bc6bdec,  
    name: 'ExpressJS' },  
  { tags: [ 'node', 'backend' ],  
    _id: 5a68fdd7bee8ea64649c2777,  
    name: 'NodeJS' } ]
```

```
Course.find({author: "Mosh", isPublished: true})  
  .limit(10)  
  .sort({name: 1})  
  .select({name: 1, tags: 1})  
  .then([courses => console.log(courses)])
```

# QUERY DOCUMENTS

## Toán tử so sánh:

Toán tử	Ý nghĩa
eq	equal
ne	not equal
gt	greater than
gte	greater than or equal

Toán tử	Ý nghĩa
lt	less than
lte	less than or equal
in	in
nin	not in

# QUERY DOCUMENTS

Ví dụ: sử dụng toán tử so sánh **gte** và **lte**

```
Course.find({price: { $gte: 10, $lte: 20 }})
  .limit(10)
  .sort({name: 1})
  .select('name tags price')
  .then(courses => console.log(courses))
```



```
[ { tags: [ 'aspnet', 'backend' ],
  _id: 5a68fde3f09ad7646ddec17e,
  name: 'ASP.NET MVC',
  price: 15 },
  { tags: [ 'angular', 'frontend' ],
  _id: 5a6900fff467be65019a9001,
  name: 'Angular',
  price: 15 },
  { tags: [ 'express', 'backend' ],
  _id: 5a68fdc3615eda645bc6bdec,
  name: 'ExpressJS',
  price: 10 },
  { tags: [ 'node', 'backend' ],
  _id: 5a68fdd7bee8ea64649c2777,
  name: 'NodeJS',
  price: 20 },
  { tags: [ 'node', 'backend' ],
  _id: 5a68fe2142ae6a6482c4c9cb,
  name: 'NodeJS by Jack',
  price: 12 },
  { tags: [ 'node', 'backend' ],
  _id: 5a68ff090c553064a218a547,
  name: 'NodeJS by Mary',
  price: 12 } ]
```



# QUERY DOCUMENTS

Ví dụ: sử dụng toán tử so sánh in

```
Course.find({ price: {$in: [10, 15, 20] } })  
  .limit(10)  
  .sort({name: 1})  
  .select('name tags price')  
  .then(courses => console.log(courses))
```

```
[ { tags: [ 'aspnet', 'backend' ],  
  _id: 5a68fde3f09ad7646ddec17e,  
  name: 'ASP.NET MVC',  
  price: 15 },  
  { tags: [ 'angular', 'frontend' ],  
  _id: 5a6900fff467be65019a9001,  
  name: 'Angular',  
  price: 15 },  
  { tags: [ 'express', 'backend' ],  
  _id: 5a68fdc3615eda645bc6bdec,  
  name: 'ExpressJS',  
  price: 10 },  
  { tags: [ 'node', 'backend' ],  
  _id: 5a68fdd7bee8ea64649c2777,  
  name: 'NodeJS',  
  price: 20 } ]
```

# QUERY DOCUMENTS

Toán tử logic:

- `.and()`

```
Course.find()  
  .and([ {author: 'Mosh'}, {isPublished: true} ])  
  .limit(10)  
  .sort({name: 1})  
  .select('name author isPublished')  
  .then([courses => console.log(courses)])
```

```
[ { _id: 5a68fde3f09ad7646ddec17e,  
  name: 'ASP.NET MVC',  
  author: 'Mosh',  
  isPublished: true },  
  { _id: 5a6900fff467be65019a9001,  
  name: 'Angular',  
  author: 'Mosh',  
  isPublished: true },  
  { _id: 5a68fdc3615eda645bc6bdec,  
  name: 'ExpressJS',  
  author: 'Mosh',  
  isPublished: true },  
  { _id: 5a68fdd7bee8ea64649c2777,  
  name: 'NodeJS',  
  author: 'Mosh',  
  isPublished: true } ]
```

# QUERY DOCUMENTS

Toán tử logic:

- `.or()`

```
Course.find()  
  .or([{author: 'Mosh'}, {isPublished: true}])  
  .limit(10)  
  .sort({name: 1})  
  .select('name author isPublished')  
  .then([courses => console.log(courses)])
```



```
[ { _id: 5a68fde3f09ad7646ddec17e,  
  name: 'ASP.NET MVC',  
  author: 'Mosh',  
  isPublished: true },  
  { _id: 5a6900fff467be65019a9001,  
  name: 'Angular',  
  author: 'Mosh',  
  isPublished: true },  
  { _id: 5a68fdc3615eda645bc6bdec,  
  name: 'ExpressJS',  
  author: 'Mosh',  
  isPublished: true },  
  { _id: 5a68fdd7bee8ea64649c2777,  
  name: 'NodeJS',  
  author: 'Mosh',  
  isPublished: true },  
  { _id: 5a68fe2142ae6a6482c4c9cb,  
  name: 'NodeJS by Jack',  
  author: 'Jack',  
  isPublished: true },  
  { _id: 5a68fdf95db93f6477053ddd,  
  name: 'React',  
  author: 'Mosh',  
  isPublished: false } ]
```

# QUERY DOCUMENTS

## Regular expression:

- Tìm những course có **name** bắt đầu với chữ **Node**

```
Course.find({name: /^Node/})  
  .limit(10)  
  .sort({name: 1})  
  .select('name author isPublished')  
  .then([courses => console.log(courses)])
```

```
[ { _id: 5a68fdd7bee8ea64649c2777,  
  name: 'NodeJS',  
  author: 'Mosh',  
  isPublished: true },  
  { _id: 5a68fe2142ae6a6482c4c9cb,  
  name: 'NodeJS by Jack',  
  author: 'Jack',  
  isPublished: true },  
  { _id: 5a68ff090c553064a218a547,  
  name: 'NodeJS by Mary',  
  author: 'Mary',  
  isPublished: false } ]
```



# QUERY DOCUMENTS

## Regular expression:

- Tìm những course có **name** kết thúc với chữ **JS**

```
[ { _id: 5a68fdc3615eda645bc6bdec,
  name: 'ExpressJS',
  author: 'Mosh',
  isPublished: true },
  { _id: 5a68fdd7bee8ea64649c2777,
  name: 'NodeJS',
  author: 'Mosh',
  isPublished: true } ]
```

```
Course.find({name: /JS$/i})
  .limit(10)
  .sort({name: 1})
  .select('name author isPublished')
  .then(courses => console.log(courses))
```

# QUERY DOCUMENTS

## Regular expression:

- Tìm những course có **name** có chứa chữ **JS**

```
Course.find({name: /. *JS. */i})  
  .limit(10)  
  .sort({name: 1})  
  .select('name author isPublished')  
  .then([courses => console.log(courses)])
```

```
[ { _id: 5a68fdc3615eda645bc6bdec,  
    name: 'ExpressJS',  
    author: 'Mosh',  
    isPublished: true },  
  { _id: 5a68fdd7bee8ea64649c2777,  
    name: 'NodeJS',  
    author: 'Mosh',  
    isPublished: true },  
  { _id: 5a68fe2142ae6a6482c4c9cb,  
    name: 'NodeJS by Jack',  
    author: 'Jack',  
    isPublished: true },  
  { _id: 5a68ff090c553064a218a547,  
    name: 'NodeJS by Mary',  
    author: 'Mary',  
    isPublished: false } ]
```

# QUERY DOCUMENTS

**.count():** đếm số lượng documents

**Chú ý:** .count() sẽ không còn trong những phiên bản sau và sẽ được thay thế bởi **.countDocuments()** hoặc **.estimatedDocumentCount()**

```
Course.find({name: /. *JS.*/i})  
  .limit(10)  
  .sort({name: 1})  
  .select('name author isPublished')  
  .count()  
  .then([courses => console.log(courses)])
```

# QUERY DOCUMENTS

**.skip():** bỏ qua một số document --> dùng để thực hiện chức năng **phân trang**

```
// /api/courses?pageNumber=2&pageSize=10
```

```
const pageNumber = 1;  
const pageSize = 3;
```

Sau này, khi viết API, `pageNumber` và `pageSize` sẽ được lấy từ `queryString`

```
Course.find()  
  .skip((pageNumber - 1)*pageSize)  
  .limit(pageSize)  
  .sort({name: 1})  
  .select('name author isPublished')  
  .then(courses => console.log(courses))
```

```
[ { _id: 5a68fde3f09ad7646ddec17e,  
  name: 'ASP.NET MVC',  
  author: 'Mosh',  
  isPublished: true },  
  { _id: 5a6900fff467be65019a9001,  
    name: 'Angular',  
    author: 'Mosh',  
    isPublished: true },  
  { _id: 5a68fdc3615eda645bc6bdec,  
    name: 'ExpressJS',  
    author: 'Mosh',  
    isPublished: true } ]
```

# QUERY DOCUMENTS

## BÀI TẬP TẠI LỚP:

1. Tìm tất cả các khóa **backend** và **đã được publish**, sort theo **name**, chỉ hiển thị **name** và **author**
2. Tìm tất cả các khóa **backend** và **frontend**, **đã được publish**, sort theo **price giảm dần**, chỉ hiển thị **name** và **author**
3. Tìm tất cả các khóa **đã được published** có giá từ \$15 trở lên và có từ **"by"** trong **name**



# QUERY DOCUMENTS

Một số phương thức tìm kiếm khác:

- **.findById():** tìm kiếm theo id (ObjectId)
- **.findOne():** tìm kiếm và chỉ trả về document đầu tiên tìm thấy, nếu không tìm được sẽ trả về **null**



# UPDATE DOCUMENT(S)

- Sử dụng **findById()**: tìm document theo id, sau đó update và save document

```
Course.findById("5bf617ac59d19db53ccb6e1f")  
  .then(course => {  
    course.author = "Hackagon"  
    course.save()  
    .then(course => console.log(course));  
  })
```



# UPDATE DOCUMENT(S)

- Sử dụng **findByIdAndUpdate()**, **findOneAndUpdate()**: các phương thức này sẽ bị **remove** trong các phiên bản sau.

```
Course.findByIdAndUpdate("5bf617ac59d19db53ccb6e1f", {
  $set: { isPublished: true }
})
.then(console.log)
```

- Xem thêm các toán tử update (update operators) tại:

<https://docs.mongodb.com/manual/reference/operator/update/>





# UPDATE DOCUMENT(S)

- Sử dụng **updateOne()**:

```
Course.updateOne({ _id: "5bf617ac59d19db53ccb6e1f"}, {  
  $set: {  
    name: "FullstackJS",  
  }  
})  
.then(console.log)
```

# UPDATE DOCUMENT(S)

- `updateMany()`:

```
Course.updateMany({  
  $set: {  
    isPublished: false  
  }  
})  
.then(console.log)
```

Update tất cả field `isPublished` thành `false` trong tất cả documents

```
Course.updateMany({  
  $unset: { isPublished: 1 }  
})  
.then(console.log)
```

Xóa field `isPublished` trong tất cả các documents

# DELETE DOCUMENT(S)

- **deleteOne()**: delete document đầu tiên thỏa mãn điều kiện

```
Course.deleteOne({author: "Mosh"})  
  .then(console.log)
```

- **deleteMany()**: delete tất cả documents thỏa mãn điều kiện

```
Course.deleteMany({author: "Mosh"})  
  .then(console.log)
```

# DELETE DOCUMENT(S)

- **findByIdAndRemove()**: delete document theo id. Phương thức này sẽ bị **remove** trong các phiên bản sau, thay thế bằng **findOneAndRemove()**

```
Course.findByIdAndRemove("5bf617ac59d19db53ccb6e23")  
    .then(console.log)
```

# BÀI TẬP VỀ NHÀ

- Link: <https://www.w3resource.com/mongodb-exercises/#PracticeOnline>
- Download file restaurants.zip và thực hiện 32 bài queries theo yêu cầu
- Nộp bài theo deadline.

